

日本国特許庁
JAPAN PATENT OFFICE

JP 20000304
#2/19/02
C. McKing
J1021 U.S. PTO
10/023147
12/18/01

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office

出願年月日
Date of Application:

2000年12月19日

出願番号
Application Number:

特願2000-386069

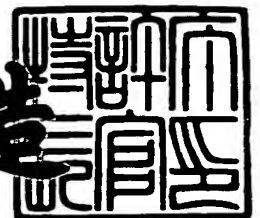
出願人
Applicant(s):

インターナショナル・ビジネス・マシーンズ・コーポレーション

2001年 6月19日

特許庁長官
Commissioner,
Japan Patent Office

及川耕造



出証番号 出証特2001-3057764

【書類名】 特許願

【整理番号】 JP9000304

【提出日】 平成12年12月19日

【あて先】 特許庁長官殿

【国際特許分類】 G06F 9/30

【発明者】

【住所又は居所】 神奈川県大和市下鶴間1623番地14 日本アイ・ピー・エム株式会社 東京基礎研究所内

【氏名】 高野 光司

【発明者】

【住所又は居所】 神奈川県大和市下鶴間1623番地14 日本アイ・ピー・エム株式会社 東京基礎研究所内

【氏名】 佐藤 証

【特許出願人】

【識別番号】 390009531

【氏名又は名称】 インターナショナル・ビジネス・マシーンズ・コーポレーション

【代理人】

【識別番号】 100086243

【弁理士】

【氏名又は名称】 坂口 博

【代理人】

【識別番号】 100091568

【弁理士】

【氏名又は名称】 市位 嘉宏

【代理人】

【識別番号】 100106699

【弁理士】

【氏名又は名称】 渡部 弘道

【復代理人】

【識別番号】 100112520

【弁理士】

【氏名又は名称】 林 茂則

【選任した復代理人】

【識別番号】 100110607

【弁理士】

【氏名又は名称】 間山 進也

【手数料の表示】

【予納台帳番号】 091156

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【包括委任状番号】 9706050

【包括委任状番号】 9704733

【包括委任状番号】 0004480

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 演算回路および演算方法

【特許請求の範囲】

【請求項 1】 複数のレジスタと、前記複数のレジスタに入力される値を入力とする演算器と、複数のメモリと、を含み、

前記複数のメモリから前記複数のレジスタへの複数の変数の読出しを、前記演算器のパイプライン処理における同一の読出しステージで行う演算回路。

【請求項 2】 前記演算器は、第 1 レジスタ、第 2 レジスタ、第 3 レジスタ、第 4 レジスタの各々に入力された r ビット長を有する x_1 、 x_2 、 x_3 、 x_4 の各入力値に基づいて、 $2r$ または $2r+1$ ビット長を有する $x_1 + x_2 \cdot x_3 + x_4$ の演算結果 Q を与える積和演算器である請求項 1 記載の演算回路。

【請求項 3】 前記複数のメモリには、第 1 メモリおよび第 2 メモリを含み、

前記パイプライン処理の演算ステージに続く演算結果の書き込みステージにおいて、前記演算結果 Q の下位 r ビット Q_L が前記第 1 メモリに記録され、前記演算結果 Q の前記 Q_L を除く上位ビット Q_H が前記第 4 レジスタに入力され、

前記書き込みステージに続く前記レジスタへの変数の読出しステージにおいて、前記第 1 メモリから前記第 1 レジスタに変数 x_1 が、前記第 2 メモリから前記第 3 レジスタに変数 x_3 が、同一の読出しステージで読み出される請求項 2 記載の演算回路。

【請求項 4】 前記第 1 および第 2 メモリは、データの書き込みポートと読出しポートとを各々 1 つ有する 2 ポートメモリである請求項 3 記載の演算回路。

【請求項 5】 前記第 1 メモリはデータの書き込みポートと読出しポートとを各々 1 つ有する 2 ポートメモリであり、前記第 2 メモリはデータの書き込みおよび読出しが 1 つのポートで行われる 1 ポートメモリである請求項 3 記載の演算回路。

【請求項 6】 前記演算器は、第 1 レジスタ、第 2 レジスタ、第 3 レジスタ、第 4 レジスタ、第 5 レジスタ、第 6 レジスタの各々に入力された r ビット長を有する x_1 、 x_2 、 x_3 、 x_4 、 x_5 、 x_6 の各入力値に基づいて、 $2r$ または

2r+1ビット長を有する $x_1 + x_2 \cdot x_3 + x_4 \cdot x_5 + x_6$ の演算結果Qを与える積和演算器である請求項1記載の演算回路。

【請求項7】 前記複数のメモリには、第1メモリ、第2メモリおよび第3メモリを含み、

前記パイプライン処理の演算ステージに続く演算結果の書き込みステージにおいて、前記演算結果Qの下位rビット Q_L が前記第1メモリに記録され、前記演算結果Qの前記 Q_L を除く上位ビット Q_H が前記第6レジスタに入力され、

前記書き込みステージに続く前記レジスタへの変数の読出しステージにおいて、前記第1メモリから前記第1レジスタに変数 x_1 が、前記第2メモリから前記第3レジスタに変数 x_3 が、前記第3メモリから前記第5レジスタに変数 x_5 が、同一の読出しステージで読み出される請求項6記載の演算回路。

【請求項8】 前記第1メモリはデータの書き込みポートと読出しポートとを各々1つ有する2ポートメモリであり、前記第2および第3メモリは、データの書き込みおよび読出しが1つのポートで行われる1ポートメモリである請求項7記載の演算回路。

【請求項9】 複数の入力レジスタを有する演算器と複数のメモリとを備えた演算回路を用いた演算方法であって、

前記入力レジスタに入力されている値に基づいて演算を行うステップと、

前記演算の結果を前記入力レジスタまたは前記メモリに書き込むステップと、

前記複数のメモリから前記複数の入力レジスタに複数の変数を同一のパイプラインステージで読み出すステップと、

を含む演算方法。

【請求項10】 前記演算器は、第1レジスタ、第2レジスタ、第3レジスタ、第4レジスタの各々に入力されたrビット長を有する x_1 、 x_2 、 x_3 、 x_4 の各入力値に基づいて、2rまたは2r+1ビット長を有する $x_1 + x_2 \cdot x_3 + x_4$ の演算結果Qを与える積和演算器である請求項9記載の演算方法。

【請求項11】 前記複数のメモリには、第1メモリおよび第2メモリを含み、

前記演算結果Qの下位rビット Q_L が前記第1メモリに記録され、前記演算結

果 Q の前記 Q_L を除く上位ビット Q_H が前記第4レジスタに入力される前記演算器のパイプライン処理における書き込みステップと、

前記第1メモリから前記第1レジスタへの変数 x_1 の読出しと、前記第2メモリから前記第3レジスタへの変数 x_3 の読出しとが前記パイプライン処理の同一の読出しステージで行われる読出しステップと、

を含む請求項10記載の演算方法。

【請求項12】 前記第1および第2メモリは、データの書き込みポートと読出しポートとを各々1つ有する2ポートメモリである請求項11記載の演算方法。

【請求項13】 前記第1メモリはデータの書き込みポートと読出しポートとを各々1つ有する2ポートメモリであり、前記第2メモリはデータの書き込みおよび読出しが1つのポートで行われる1ポートメモリである請求項11記載の演算方法。

【請求項14】 前記演算器は、第1レジスタ、第2レジスタ、第3レジスタ、第4レジスタ、第5レジスタ、第6レジスタの各々に入力された r ビット長を有する x_1 、 x_2 、 x_3 、 x_4 、 x_5 、 x_6 の各入力値に基づいて、 $2r$ または $2r+1$ ビット長を有する $x_1 + x_2 \cdot x_3 + x_4 \cdot x_5 + x_6$ の演算結果 Q を与える積和演算器である請求項9記載の演算回路。

【請求項15】 前記複数のメモリには、第1メモリ、第2メモリおよび第3メモリを含み、

前記演算結果 Q の下位 r ビット Q_L が前記第1メモリに記録され、前記演算結果 Q の前記 Q_L を除く上位ビット Q_H が前記第6レジスタに入力される前記演算器のパイプライン処理における書き込みステップと、

前記第1メモリから前記第1レジスタへの変数 x_1 の読出しと、前記第2メモリから前記第3レジスタへの変数 x_3 の読出しと、前記第3メモリから前記第5レジスタへの変数 x_5 の読出しとが、前記パイプライン処理の同一の読出しステージで行われる読出しステップと、

を含む請求項14記載の演算方法。

【請求項16】 前記第1メモリはデータの書き込みポートと読出しポート

とを各々1つ有する2ポートメモリであり、前記第2および第3メモリはデータの書き込みおよび読出しが1つのポートで行われる1ポートメモリである請求項15記載の演算方法。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】

本発明は、演算回路および演算方法に関し、特に公開鍵暗号化方式等に好適な乗剰余計算の高速化に関するものである。

【0002】

【従来の技術】

情報を伝送する際に、セキュリティの確保あるいは認証のために公開鍵暗号（非対称暗号）が用いられる。公開鍵暗号は公開鍵と秘密鍵の一对の鍵を用いて情報を伝送する暗号方式である。送信者が受信者の公開鍵で平文を暗号化し、暗号文を受取った受信者は受信者しか知りえない秘密鍵を用いて暗号文を復号する。このような公開鍵暗号では共通鍵暗号（対称鍵暗号）のように1つの共通鍵を共有する必要がなく、また、公開鍵暗号では公開鍵を広く公開することができるので不特定多数の者との通信の秘密が確保できる。さらに、公開鍵暗号を電子認証あるいは電子署名に用いて面識のない他人との信頼関係を確立することができる。インターネット等通信技術が支えるネットワーク社会あるいはその中で営まれる商取引等において必須の技術と位置付けられる。

【0003】

公開鍵暗号方式の一つにRSAが知られている。RSAは非常に大きな整数の離散対数問題あるいは素因数分解の困難性にその安全性の基礎をおく。たとえば、公開鍵（ e , n ）を用いて、平文 M を

$$C = M^e \pmod{n}$$

の関係式から暗号文 C を生成する（なお、 M は整数 n 未満になるようブロック化されている）。この暗号文 C の解読には離散対数計算（ a , y , p から $y = a^x \pmod{p}$ となる x を見出す）が必要となり

$$O(2^{\sqrt{\log n}})$$

の計算量を必要とする (S Q R T は二乗根を与える関数である)。整数 n が少なくとも 512 ビット長以上、好ましくは 1024 ビット長以上であれば実用的な計算時間での解読は困難になる。

【0004】

ところが、公開鍵 (e, n) と

$e d \pmod{\text{lcm}(p-1, q-1)} = 1$, $n = p q$ 、(ただし p, q は十分大きな素数)

の関係にある秘密鍵 (d, n) を用いれば、

$$M = C^d \pmod{n}$$

の関係式を用いて平文 M を簡単に求めることができる (ただし、 $\text{lcm}(a, b)$ は a と b の最小公倍数を与える)。

【0005】

たとえば、 $d = 11$ の場合、

$$C^{11} = ((C^2)^2 C)^2 C$$

のように d を 2 進展開して、自乗剰余演算と乗算剰余演算を繰り返せば、高々 d のビット長の 2 倍の回数の乗剰余演算で計算できる。

【0006】

しかしながら、上記べき乗剰余演算であっても、たとえば D E S (data encryption standard) 等の対称暗号に比較すれば計算量が多くなる。このためできるだけ効率的なアルゴリズムとその実装が求められる。

【0007】

上記べき乗剰余演算における自乗剰余演算と乗算剰余演算を高速化する手法に、たとえば Peter L. Montgomery 著、「Modular Multiplication Without Trial Division」、Mathematics of computations, Vol.44, No.170 April 1985, pp 519-522、に記載されているモンゴメリ乗算手法がある。モンゴメリ乗算は乗剰余算を、加算、乗算、シフト演算の繰り返しにより、減算を繰り返す除算よりも少ない計算量で実現する手法である。以下にモンゴメリ乗算の計算主要部

$$P \equiv X Y R^{-1} \pmod{n}$$

を擬似コード 1. x に示す。なお、上式において、

$$R = (2^r)^m$$

$$N \equiv -n^{-1} \pmod{2^r}$$

とする。また、擬似コードにおいて各行の左側には行番号を付す（以下同様）。

【0008】

(1.1) $P = 0;$

(1.2) $\text{for } (i=0 ; i < m ; i++) \{$

(1.3) $t = (p_0 + x_i y_0) N \pmod{2^r};$

(1.4) $P = (P + x_i Y + t \cdot n) / 2^r;$

(1.5) $\};$

(1.6) $\text{if } (P \geq n) P = P - n;$

上記擬似コード1. xに示すように、その主要部分の繰り返し演算は次のようになる。まず、Xをm個のブロック x_i に区切り（ $X = (x_{m-1}, x_{m-2}, \dots, x_1, x_0)$ ）、Yとの部分積加算（ $x_i Y$ ）をm回繰り返す（行番号1. 2～1. 5）。このとき、途中結果Pの最下位ブロック p_0 が0となるようなnの倍数「 $t \cdot n$ 」を毎回加算する（行番号1. 4）。tは行番号1. 3において定義されている。さらに、Pをrビットだけ右にシフト、つまり「 2^{-r} 」を乗ずる（行番号1. 4）。なお、rビットのシフト演算は、m回のシフト演算により $2^{-r m} = R^{-1}$ となって R^{-1} の乗算をすることになる。

【0009】

たとえば、512ビットのモンゴメリ乗算を32ビット乗算器によって実行すると仮定すると、 $m = 512 / 32 = 16$ 回のループを繰り返すことになる。上記擬似コードでは簡単のために $x_i \cdot Y$ や、 $t \cdot n$ のように32ビット×512ビットのように示したが、実際は512ビットのYとnも16個の32ビットブロックに分割して計算を実行する。つまり、Pの部分積加算は演算において $m = 16$ の二重ループとなる。以下に二重ループによってモンゴメリ乗算を実行する手順の一例を擬似コード2. xに示す。

【0010】

(2.1) $P = 0;$

(2.2) $\text{for } (i=0 ; i < m ; i++) \{$

```

(2.3)       $t = p_0 + x_i y_0 \pmod{2^r};$ 
(2.4)       $t = t \cdot N \pmod{2^r};$ 
(2.5)       $c = 0;$ 
(2.6)      for (j=0 ; j < m ; j++) {
(2.7)           $tmp = p_j + x_i \cdot y_j + c;$ 
(2.8)           $tmp = tmp + t \cdot n_j;$ 
(2.9)          if (j != 0)  $p_{j-1} = tmp \pmod{2^r};$ 
(2.10)          $c = tmp / 2^r;$ 
(2.11)     };
(2.12)      $p_{m-1} = c;$ 
(2.13)     };
(2.14)     if (P => n) P = P-n;

```

なお、ここで、X, Y, nはm個のブロックに分割されている。つまり、

$$X = (x_{m-1}, x_{m-2}, \dots, x_1, x_0)$$

$$Y = (y_{m-1}, y_{m-2}, \dots, y_1, y_0)$$

$$n = (n_{m-1}, n_{m-2}, \dots, n_1, n_0)$$

乗算器が一つの場合を仮定すると、途中結果tmpの計算に二回の積和演算が必要となる。変数 p_j , x_i , y_j , t , n_j は全てrビット長の数、変数cは下位ブロックからのキャリーである。上記擬似コード2. xの例では、一回のjループで2rビットの数 $x_i \cdot y_j$ と $t \cdot n_j$ 、そしてr+1ビットの数 p_j とcの加算を行い(行番号2. 6~2. 11)、積和演算後の途中結果tmpは2r+1ビット長を持つ。tmpの下位rビットは変数 p_j に、上位r+1ビットは変数cにストアされる(行番号2. 9、2. 10)。

【0011】

一方、 $x_i \cdot y_j$ と $t \cdot n_j$ の加算を2つの別のループで行うことも可能である。この例を擬似コード3. xに示す。

【0012】

```

(3.1)      P = 0;
(3.2)      for (i=0 ; i < m ; i++) {

```

```

(3.3)      c = 0;
(3.4)      for (j=0 ; j < m ; j++) {
(3.5)          tmp = pj + xi·yj + c;
(3.6)          pj = tmp(mod 2r);
(3.7)          c = tmp/2r;
(3.8)      };
(3.9)      pm = c; c = 0;
(3.10)     t = p0·N (mod 2r);
(3.11)     for (j=0 ; j < m ; j++) {
(3.12)         tmp = pj + t·nj + c;
(3.13)         if (j != 0) pj-1 = tmp(mod 2r);
(3.14)         c = tmp/2r;
(3.15)     };
(3.16)     pm-1 = pm + c;
(3.17) };
(3.18) if (P => n) P = P-n;

```

擬似コード 3. x の例では、変数 p_j は r ビット長、変数 tmp は $2r$ ビット長となる。

【 0 0 1 3 】

擬似コード 2. x、3. x の両例ともモンゴメリ乗算の二重ループを抜けた後の結果 P は $2n$ 未満となるが、 n 以下である保証はないので必要に応じて、「 $P = P - n$ 」とする（行番号 2. 14, 3. 18）。

【 0 0 1 4 】

上記擬似コード 2. x、3. x の例において、基本的に p_j は、 P を m 個に分割した r ビットレジスタで正の値をとる。しかし、擬似コード 2. x の例では、行番号 2. 12 の「 $p_{m-1} = c$ 」において、 c は最大 $r+1$ ビットとなる。この時、演算ビット数が r ビットで割り切れる場合には 1 ビットあふれることになる。よって、ループ回数を $m+1$ にするか、ループ回数を m 回に押さえるための最上位ビットの特別な処理が必要となる。本明細書では簡単のため上記例の両者

とも j ループは同じループ回数 m であるものとする。なお両例の行番号 2. 14, 3. 18 の比較演算「if ($P \geq n$)」において、符号ビットの考慮などを行う必要がある。

【0015】

【発明が解決しようとする課題】

上記擬似コード 2. x、3. x の例により、モンゴメリ乗算の計算を実行することができる。しかし、本発明者らがその演算サイクルを詳細に検討したところ、演算のパフォーマンスは変数のレジスタ割り付けやメモリ構成に大きく影響されることが判明した。以下検討結果について説明する。

【0016】

上記擬似コード 2. x の場合、 r ビット長の変数 N は i ループ（擬似コード 2. 2～2. 13）内で値が変わらずに繰り返し使用される。また、 t と x_i は j ループ（擬似コード 2. 6～2. 11）内で値が変わらずに繰り返し使用される。このため、変数 N , t , x_i については、一旦計算あるいはメモリから読み出した後は各ループを抜けるまでレジスタに保持して繰り返し使用することができる。また j ループ内で使用する途中変数 t_{mp} と c はレジスタに直接割り付けられる。その他の変数は、読み出しと書き込みが独立したアドレスに対して実行できる一般的な 2ポートメモリに記録する。またメモリ読み出し→演算→書き込みの一連の処理はパイプライン処理されるものとする。そうすると、演算と書き込みは基本的に 1 サイクルで終了するのでメモリ読み出しがパフォーマンスのボトルネックになる。以下、メモリ読み出しのサイクル数を検討する。

【0017】

まず行番号 2. 1 の「 $P = 0$ 」の処理については、行番号 2. 7 の演算

$$t_{mp} = p_j + x_i \cdot y_j + c$$

などで最初に p_j を読み出すときにその値をリセットすればよい。このためサイクル数 0 である。

【0018】

行番号 2. 3, 2. 4 の変数 t の計算

$$t = p_0 + x_i y_0 \pmod{2^r}$$

$$t = t \cdot N \pmod{2^r}$$

ではレジスタに割り付けられているN以外の変数 (p_0 、 x_i 、 y_0) をメモリから読み出すのに3サイクルを要する。なお、 t については前記した通り行番号2. 3の演算結果が直接レジスタに割り付けられるので読出しの必要はない。行番号2. 5の「 $c = 0$ 」は、「 $P = 0$ 」の処理と同様0サイクルである。

【0019】

j ループに入って、行番号2. 7の演算

$$tmp = p_j + x_i \cdot y_j + c$$

は p_j と y_j の読み出しで2サイクルを必要とする。 x_i は行番号2. 3の t の計算の際に読み出されているのでここでの読出しの必要はない。また、 c については前記の通り直接レジスタに割り付けられるので読出しの必要はない。

【0020】

行番号2. 8の演算

$$tmp = tmp + t \cdot n_j$$

は n_j の読み出しで1サイクルを要する。 tmp 、 t については前記した通り直接レジスタに割り付けられるので読出しの必要はない。

【0021】

行番号2. 9の演算

$$p_{j-1} = tmp \pmod{2^r}$$

$$c = tmp / 2^r$$

の実際のハードウェア上の動作は、「 $tmp + t \cdot n_j$ 」を一旦レジスタ tmp で受けずに直接 p_{j-1} と c に書き込めばよい。よってどちらも0サイクルとなる。

【0022】

j ループを抜けた直後の行番号2. 12の演算

$$p_{m-1} = c$$

はパイプライン処理により0サイクルとなる。

【0023】

したがって i ループ内の演算サイクル数は、

$$3 + m(2 + 1) = 3m + 3$$

となる。これに i ループの m を乗じて

$$m(3m + 3) = 3m^2 + 3m$$

となる。しかし、先に述べたように $i = 0$ のとき $p_j = 0$ なので p_j を読み出す必要はなく、入力を 0 リセットすればよい。よって i ループ全体ではこの分の m を減じて

$$3m^2 + 3m - m = 3m^2 + 2m$$

となる。行番号 2. 14 の比較動作

$$if \quad (P \geq n)$$

では減算「 $P - n$ 」によって大小を判定するのでこれに 2 m サイクル、最後にパイプラインを抜けるのに 2 サイクルを要する。結局最終的なサイクル数は

$$3m^2 + 4m + 2$$

となる。これはパイプライン処理が乱れなく実行できる場合であり、演算ビット数が乗算器のビット数に対してあまり大きくないとき、つまりブロック数 m が少ないときは、書き込もうとした値を直ぐに読み出す必要があるためメモリアクセス待ちのオーバーヘッドが加わることになる。ただしブロック数が 4 ~ 5 あれば実用上このような問題は生じない。また逆にブロック数が 2 ~ 3 といった小さな数に対してモンゴメリ法を適用する意味はあまりないので、上記事情は考慮する必要がない。つまり、二重ループによる m^2 項の係数「3」が大きく変わることはない。また、実装形態によっては j ループ前後でパイプライン動作が乱れることがあり、前記最終的なサイクル数が実際には若干異なることがある。しかし、この場合であっても二重ループによる m^2 項の係数「3」は変わらず、最終的なサイクル数が前記値と大きく異なることはない。

【0024】

これと同様に擬似コード 3. x の例を詳細に検討すれば、以下の通りである。

行番号 3. 1 の「 $P = 0$ 」の処理は、行番号 3. 5 の演算

$$tmp = p_j + x_i \cdot y_j + c$$

で最初に p_j を読み出すときにその値を 0 リセットすればよいのでサイクル数は 0 である。行番号 3. 1 の「 $c = 0$ 」の処理についても、擬似コード 2. x と同

様0サイクルである。

【0025】

第1 j ループ (行番号3. 4~3. 8) に入って、行番号3. 5の演算

$$t m p = p_j + x_i \cdot y_j + c$$

では、 p_j と y_j の読み出しで2サイクルを必要とし、第1 j ループの最初に x_i を読み出す必要がある。行番号3. 6の演算

$$p_j = t m p \pmod{2^r}$$

$$c = t m p / 2^r$$

は擬似コード2. xの場合と同様に $t m p$ を直接 p_j 、 c に書き込むので0サイクルとなる。よって、1番目の j ループのサイクル数は、 $2m+1$ となる。

【0026】

第1 j ループを抜けた後の行番号3. 9の演算

$$p_m = c; \quad c = 0;$$

はパイプライン処理により0サイクルとなり、行番号3. 10の演算

$$t = p_0 \cdot N \pmod{2^r}$$

では、 p_0 と N の読み出しに2サイクルを要する。

【0027】

次に第2 j ループ (行番号3. 11~3. 15) に入って、行番号3. 12の演算

$$t m p = p_j + t \cdot n_j + c$$

では、 p_j と n_j の読み出しで2サイクルを要する。なお、 t と c はレジスタに直接書込まれるので読出しの必要はない。それに続く行番号3. 13の演算

$$p_{j-1} = t m p \pmod{2^r}$$

$$c = t m p / 2^r$$

は、擬似コード2. xの場合と同様0サイクルとなる。従って、2番目の j ループのサイクル数は $2m$ となる。そして2番目の j ループを抜けた後の行番号3. 16の演算

$$p_{m-1} = p_m + c$$

では、 p_m の読出しに1サイクル必要である。

【0028】

よって、i ループ内の演算サイクル数は、

$$(2m+1) + 2 + 2m+1 = 4m+4$$

となる。これに i ループの m を乗じて

$$m(4m+4) = 4m^2 + 4m$$

となる。しかし、先に述べたように $i = 0$ のとき p_j を読み出す必要はなく、入力を 0 リセットすればよいので、i ループ全体ではこの分の m を減じて

$$4m^2 + 4m - m = 4m^2 + 3m$$

となる。行番号 3. 18 の比較動作

$$if \quad (P > n)$$

では減算「 $P - n$ 」によって大小を判定するのでこれに 2 m サイクル、最後にパイプラインを抜けるのに 2 サイクルを要する。結局最終的なサイクル数は

$$4m^2 + 5m + 2$$

となる。擬似コード 3. x の例で m^2 サイクル多いのは、tmp への部分積加算を一つの j ループで行わないため、行番号 3. 4 ~ 3. 8 のループでメモリに書き込んだ p_j を行番号 3. 11 から 3. 15 のループで読み出すためである。サイクル数こそ多いものの桁上がりが 1 ビット少なく r ビットブロックにおさまリ、レジスタに保持すべき変数も少ないという擬似コード 2. x の例と比較したメリットがある。このため、制御と回路規模の点では 3. x の例の方が有利である。図 7 に擬似コード 3. x の場合のモンゴメリ乗算回路データパス部の概略を、また図 8 にこの回路におけるブロック数 $m = 4$ の最終ループ処理近辺のタイミングチャートを示す。なお、パイプライン処理の乱れ等により、前記最終的な読出しサイクル数が実際には若干変化することがあるのは擬似コード 2. x の場合と同様である。

【0029】

上記のメモリ読出しタイミングにおいてモンゴメリ乗算の高速化を考えるとすれば、二重ループの内側で乗算を 2 回行うので、単純に乗算器を 2 つ実装すれば 2 倍の速度が得られるように思える。しかし、メモリ読み出しがボトルネックとなるのは前記した通りである。このボトルネック解消の方法として、独立な読

み出しを行える読出しポートを2つ、書き込みポートを1つ持つ3ポートメモリを使用することが考えられる。しかしながら3ポートメモリにより高速化が図れるものの、3ポートメモリは一般的ではなく、使用できるテクノロジーが限定される。すなわち、回路の汎用性が損なわれてしまう。このため半導体設計における回路設計の標準化（IPコア化）の障害になる可能性がある。さらに、3ポートメモリは素子面積が大きくなり小型化の要請にそぐわない。

【0030】

本発明の目的は、モンゴメリ乗算回路におけるメモリアクセスのボトルネックを2ポートやシングルポートの汎用メモリを使いながら解消することにある。これにより高いパフォーマンスと素子の小型化を実現し、暗号生成回路を低いコストで提供することが可能になる。

【0031】

【課題を解決するための手段】

本発明の概要を説明すれば以下の通りである。すなわち、本発明の演算回路および演算方法では、1サイクルで実行する演算に必要な変数のうちメモリから読み出す必要のある変数については異なるメモリに記録し、1つの読出しステージでこれら変数を同時に読み出すことを可能にしたものである。これにより、読出しによる待ち時間を最小にして計算のパフォーマンスを向上することができる。しかも、変数を記録するメモリには3ポートメモリ等複雑な回路、大面積を要する回路を用いることなく、2ポートやシングルポートの汎用的なメモリを用いてIPコア化をしやすくし、また素子面積を最小限にすることを可能にする。

【0032】

以下本発明を列記すれば以下の通りである。すなわち、本発明の演算回路は、複数のレジスタと、前記複数のレジスタに入力される値を入力とする演算器と、複数のメモリとを有し、前記複数のメモリから前記複数のレジスタへの複数の変数の読出しを、前記演算器のパイプライン処理における同一の読出しステージで行うものである。また、本発明の演算方法は、複数の入力レジスタを有する演算器と複数のメモリとを備えた演算回路を用いた演算方法であって、前記入力レジスタに入力されている値に基づいて演算を行うステップと、前記演算の結果を前

記入レジスタまたは前記メモリに書き込むステップと、前記複数のメモリから前記複数の入力レジスタに複数の変数を同一のパイプラインステージで読み出すステップと、を含む。

【0033】

なお、前記演算器は、第1レジスタ、第2レジスタ、第3レジスタ、第4レジスタの各々に入力された r ビット長を有する x_1 、 x_2 、 x_3 、 x_4 の各入力値に基づいて、 $2r$ または $2r+1$ ビット長を有する $x_1 + x_2 \cdot x_3 + x_4$ の演算結果 Q を与える積和演算器とすることができる。この場合、前記複数のメモリには、第1メモリおよび第2メモリを含み、前記パイプライン処理の演算ステージに続く演算結果の書き込みステージにおいて、前記演算結果 Q の下位 r ビット Q_L が前記第1メモリに記録され、前記演算結果 Q の前記 Q_L を除く上位ビット Q_H が前記第4レジスタに入力され、前記書き込みステージに続く前記レジスタへの変数の読出しステージにおいて、前記第1メモリから前記第1レジスタに変数 x_1 が、前記第2メモリから前記第3レジスタに変数 x_3 が、同一の読出しステージで読み出されるものとすることができる。また、前記第1および第2メモリを、データの書き込みポートと読出しポートとを各々1つ有する2ポートメモリとすること、あるいは、前記第1メモリをデータの書き込みポートと読出しポートとを各々1つ有する2ポートメモリ、前記第2メモリをデータの書き込みおよび読出しが1つのポートで行われる1ポートメモリとすることができる。

【0034】

また、前記演算器は、第1レジスタ、第2レジスタ、第3レジスタ、第4レジスタ、第5レジスタ、第6レジスタの各々に入力された r ビット長を有する x_1 、 x_2 、 x_3 、 x_4 、 x_5 、 x_6 の各入力値に基づいて、 $2r$ または $2r+1$ ビット長を有する $x_1 + x_2 \cdot x_3 + x_4 \cdot x_5 + x_6$ の演算結果 Q を与える積和演算器とすることができる。この場合、前記複数のメモリには、第1メモリ、第2メモリおよび第3メモリを含み、前記パイプライン処理の演算ステージに続く演算結果の書き込みステージにおいて、前記演算結果 Q の下位 r ビット Q_L が前記第1メモリに記録され、前記演算結果 Q の前記 Q_L を除く上位ビット Q_H が前記第6レジスタに入力され、前記書き込みステージに続く前記レジスタへの変数の

読出しステージにおいて、前記第1メモリから前記第1レジスタに変数 x_1 が、前記第2メモリから前記第3レジスタに変数 x_3 が、前記第3メモリから前記第5レジスタに変数 x_5 が、同一の読出しステージで読み出されるものとすることができる。また、前記第1メモリをデータの書き込みポートと読出しポートとを各々1つ有する2ポートメモリとし、前記第2および第3メモリを、データの書き込みおよび読出しが1つのポートで行われる1ポートメモリとすることができる。

【0035】

【発明の実施の形態】

以下、本発明の実施の形態を図面に基づいて詳細に説明する。ただし、本発明は多くの異なる態様で実施することが可能であり、本実施の形態の記載内容に限定して解釈すべきではない。なお、実施の形態の全体を通して同じ要素には同じ番号を付するものとする。

【0036】

(実施の形態1)

図1は、本発明の一実施の形態であるモンゴメリ乗算回路の一例をそのデータパス部について示したブロック図である。本実施の形態のモンゴメリ乗算回路は積和演算回路1と、入力レジスタ2～5と、マルチプレクサ6と、2つのメモリ7（メモリA）およびメモリ8（メモリB）とを有する。

【0037】

積和演算回路1は、レジスタ3とレジスタ4の入力値（ r ビット長）の積に、レジスタ2とレジスタ5の入力値（ r ビット長）を加算して $2r$ ビット長の出力 t_{mp} を与える。すなわち、加算入力 p_j 、 c と積算入力 y_j 、 x_i とから出力 $t_{mp} = p_j + y_j \cdot x_i + c$ を得る。積和演算回路1は、たとえば公知の全加算器FAと半加算器HAとを用いて構成できる。なお、図1において $x_i(0)$ のように示しているのは、変数 x_i の括弧内の数字に相当するビットを抜き出した値に対応する。

【0038】

入力レジスタ2～5には、メモリ7、8から値が読み出されて入力され、ある

いは積和演算回路1からの出力が直接入力される。マルチプレクサ6は3つの入力に対し1つの出力を与えるスイッチである。

【0039】

メモリ7, 8は汎用的な2ポートメモリ、すなわちデータの書き込みと読出しを各々独立した1つのポートから行えるメモリである。メモリ7(メモリA)には二重ループ処理中の途中変数 $P(p_j)$ が記録され、メモリ8(メモリB)には、その他の変数 $X(x_i)$, $Y(y_j)$, $n(n_j)$, N と最終結果 $P(p_j)$ が記録される。メモリ7, 8は何れも2ポートメモリであり、素子のIPコア化の障害、素子面積の増大の問題を生じない。

【0040】

本実施の形態の演算回路では、データの読出し、演算、書き込みが一連のパイプラインとして処理される。すなわち、メモリからレジスタへのデータの読出しステージと、レジスタに入力された値に基づく演算ステージと、演算結果をレジスタまたはメモリに書き込む書き込みステージとが並列に処理される。

【0041】

図2は、図1の回路において擬似コード3. xの処理を実行した時の処理のタイミングを示したタイミングチャートである。図2では、 $m=4$ の場合の処理の最終ループ近辺を示している。なお前記および以下のタイミングチャートにおいて一連のパイプライン処理(データの読出し、演算、結果の書き込み)を同じ背景色で示すようにグレーまたは白に濃淡分けてして示している。また、チャートの左端には、読出し先のメモリ(MemA(read)、MemB(read))、レジスタ(p_j (レジスタ5)、 y_j , n_j , N (レジスタ4)、 c (レジスタ2)、 x_i , t (レジスタ3))、書き込み先のメモリ(MemA(write)、MemB(write))を指標として示している。チャートの各行には、読出しステージにおいて各メモリから読み出される値、演算ステージにおけるレジスタ内の値、書き込みステージにおけるメモリ書込まれる値が記入されている。

【0042】

擬似コード3. xの第1jループ(行番号3. 4~3. 8)における処理、たとえば図2の $i=3$ ループ中の最初の $j=0\sim3$ の処理では、まず、 $j=0$ のル

ープで、メモリ7 (MemA) からレジスタ5 (p_j) に p_0 を、メモリ8 (MemB) からレジスタ4 (y_j, n_j, N) に y_0 を読み出す (読出しステージ)。なおこの読出しステージと同時に前サイクルの演算ステージ (行番号3. 16の計算) および前々サイクルの書込みステージ (メモリ7 (MemA) への p_2 の書込み、c レジスタへの上位ビット tmp_H の書込み) が行われている。

【0043】

次に、 $j = 1$ の読出しステージでメモリ7 (MemA) からレジスタ5 (p_j) に p_1 を、メモリ8 (MemB) からレジスタ4 (y_j, n_j, N) に y_1 を読み出すと同時に先に読み込まれた p_0, y_0 とレジスタ2 (c) に入力されている前ステップの演算結果 (tmp_H) およびレジスタ3 (x_i, t) に入力されている x_3 に基づいて行番号3. 5の演算を実行する (演算ステージ)。なお、この時前サイクルの書込みステージ (メモリ7 (MemA) への p_3 の書込み、c レジスタ (レジスタ2) への上位ビット tmp_H の書込み) が行われている。

【0044】

次に、 $j = 2$ の読出しステージでメモリ7 (MemA) からレジスタ5 (p_j) に p_2 を、メモリ8 (MemB) からレジスタ4 (y_j, n_j, N) に y_2 を読み出すと同時に前サイクルでレジスタに読み出された値に基づいて演算を実行し、先の演算結果 (変数 p_0, x_3, y_0, c に基づく演算結果) の上位ビット tmp_H および下位ビット p_0 を各々c レジスタとメモリ7 (MemA) に書き込む (書込みステージ)。このようにして、 p_0, y_0 の入力以降結果が記録されるまでの一連のパイプライン処理が行われる。その他の p_j, y_j についても同様である。

【0045】

次に、行番号3. 9の処理では、読み込みサイクルに1サイクルの空きを設け、 $j = 3$ の演算ステージの後にレジスタcの内容を p_4 としてメモリ7に記録する。

【0046】

次に、行番号3. 10の処理 (t の計算) では、メモリ7から p_0 を、メモリ

8からNを同一の読出しステージで各々レジスタ3およびレジスタ4に読出し、演算の結果を次の書込みステージでレジスタ4に変数tとして書き込む。

【0047】

次に、行番号3. 11～3. 15の第2jループの処理では、第1jループの場合と同様にメモリ7から p_j が、メモリ8から n_j が同一読出しステージで読み出され、先に入力された変数t、cを用いて行番号3. 12の演算を行う。その後、演算結果の下位ビットは p_{j-1} としてメモリ7に記録され（行番号3. 13）、上位ビットはcレジスタに記録される（行番号3. 14）。なお、この第2jループの $j=0$ における処理ではcは0リセットされる。また、 $j=0$ では演算結果の下位ビットは記録されない。

【0048】

行番号3. 16の処理では、メモリ7から p_4 がレジスタ5（ p_j ）読み出され、cレジスタの値との和をとって p_3 としてメモリ7に記録される。ただし、最後のiループのメモリへの書込み処理では、 p_j はメモリ7だけでなくメモリ8にも記録する。

【0049】

最後に、行番号3. 18の処理では、メモリ7から p_j と n_j を各々レジスタに読出し、比較演算を行う。つまり減算「 $P = P - n$ 」の結果はメモリ8（MemB）の前の結果Pを上書きしない場所に保存する。メモリ8（MemB）は最後のjループを抜けた時のPと、それからnを減じた $P - n$ を保持することになるが、 $P - n$ が正であれば $P - n$ を、 $P - n$ が負であればPを最終結果とする。最後のjループを抜けた時にメモリ7（MemA）にもPを書き込むのは単に $P - n$ の計算でメモリ8（MemB）に保持されているnと同時に読み出すためである。

【0050】

上記のようにして、擬似コード3. xの演算処理を終了する。上記演算処理では、メモリからの変数の読出しを同一の読出しステージで行う。このように同一の読出しステージで変数の読出しが行えるのは、メモリを2つ設け、同時に読み出す可能性のある変数を相違するメモリに記録できるようにしたためである。こ

れにより、従来の方式に比較してメモリ読出しの待ち時間を短縮して、処理のパフォーマンスを向上できる。

【0051】

上記処理による読出しステージのサイクル数を検討すれば、以下の通りである。第1 j ループの処理で m、行番号 3. 9, 3. 10 の処理で 2、第2 j ループの処理で m、行番号 3. 16 の処理で 1、よって i ループ内の処理に必要なサイクル数は $2m + 3$ となる。これに i ループの m 回を乗じて $2m^2 + 3m$ 。なお $i = 0$ のときに p_j を読み出す必要はないがそのサイクルで y_j を読み出す必要があるので m を減じることはない。行番号 3. 18 の比較動作に m サイクル、パイプラインを抜けるのに 2 サイクルを要する。結局最終的なサイクル数は

$$2m^2 + 4m + 2$$

となる。従来方式において $4m^2 + 5m + 2$ の読出しサイクルを必要としたことと比較すれば、約半分のサイクル数に減少する。

【0052】

なお、前記実施の形態では擬似コード 3. x の場合を説明したが、擬似コード 2. x のように $x_i \cdot y_j$ と $t \cdot n_j$ を一つのループで加算することも可能である。擬似コード 2. x の場合は、擬似コード 3. x の場合に比べ変数 p_j の読み出しの待ちサイクルが半分と少ないため、全体のサイクル数が少ないというメリットがあったが、前記実施の形態のようにメモリを 2 つ設けてこれを使い分ける場合、両者に違いはない。これは図 2 からわかるように、本実施の形態では p_j と y_j または p_j と n_j の読み出しを別のメモリから同時に行え、余計な待ちが生じないためである。本実施の形態では、擬似コード 2. x の場合であっても、従来方式のサイクル数、 $3m^2 + 4m + 2$ と比較して、そのおよそ $2/3$ になる。 p_{m-1} に余計な桁上がりが生じて $r + 1$ ビットになることもないので、例外処理が不要であり、回路構成が簡単になるメリットもある。

【0053】

(実施の形態 2)

図 3 は、本発明の他の実施の形態であるモンゴメリ乗算回路の一例をそのデータパス部について示したブロック図である。本実施の形態の演算回路はメモリ 8

(メモリ B) が 1 ポートメモリ、つまりデータの読出しと書込みの双方を 1 つのポートで行うメモリとすることを除き、実施の形態 1 と同様である。このように本実施の形態ではメモリ 8 を 1 ポートメモリとすることにより、メモリサイズを小さくして回路の小型化を図ることが可能になる。

【 0 0 5 4 】

図 4 は、図 3 の回路において擬似コード 3. x の処理を実行した時の処理のタイミングを示したタイミングチャートである。図 4 では、 $m = 4$ の場合の処理の最終ループ近辺を示している。図示するように i ループが終了するまでの処理は実施の形態 1 と同様である。 $P = P - n$ の処理において、本実施の形態ではメモリ 8 に 1 ポートメモリを用いているので最終的な計算結果である p_j の書込みをメモリ 8 に行わず、メモリ 7 に行く。そして、最後にメモリ 7 (MemA) からメモリ 8 (MemB) に p_j の転送を行う。すなわち、最終結果 P の補正演算は、 $P \geq n$ ならば $P - n$ を最終結果とするものであるが、 $P \geq n$ かどうかは減算を一回実行するまで判定できない。そのため実施の形態 1 では P と $P - n$ の双方をメモリ 8 (MemB) に書き込んで、減算結果の符合を見ていずれか一方を選択するようにした。本実施の形態では P と $P - n$ をメモリ 7 (MemA) にだけ書き込み、減算結果に応じて P か $P - n$ のいずれか一方をメモリ 8 (MemB) に書き込むようにする。このようにすれば、 $m + 1$ サイクル増加するだけのペナルティーでメモリ 8 (MemB) をシングルポートメモリにすることができる。これによりメモリの小型化を図ることが可能になる。なお、本実施の形態の場合のサイクル数は

$$2m^2 + 5m + 3$$

となる。これに対して、従来のように 1 つのシングルポートメモリで擬似コード 2. x を処理した場合、

$$4m^2 + 5m + 2$$

擬似コード 3. x を処理した場合、

$$6m^2 + 7m + 2$$

のサイクル数が必要になる。本発明の優位性がより顕著となる。

【 0 0 5 5 】

(実施の形態 3)

図 5 は、本発明のさらに他の実施の形態であるモンゴメリ乗算回路の一例をそのデータパス部について示したブロック図である。本実施の形態のモンゴメリ乗算回路は積和演算回路 9 と、入力レジスタ 10～15 と、3 つのメモリ 16 (メモリ A)、メモリ 17 (メモリ B 1) およびメモリ 18 (メモリ B 2) とを有する。

【0056】

積和演算回路 9 は、レジスタ 11 (t レジスタ) とレジスタ 12 (n_j , N レジスタ) の入力値 (r ビット長) の積と、レジスタ 13 (x_i レジスタ) とレジスタ 14 (y_j レジスタ) の入力値 (r ビット長) の積と、レジスタ 10 (c レジスタ) の入力値 ($r+1$ ビット長) と、レジスタ 15 (p_j レジスタ) の入力値 (r ビット長) を加算した $2r+1$ ビット長の出力 t_{mp} を与える。すなわち、積和演算回路 9 は 2 つの乗算器を有し、加算入力 p_j , c と積算入力 y_j , x_i と、積算入力 t , n_j とから出力 $t_{mp} = p_j + y_j \cdot x_i + t \cdot n_j + c$ を得る。

【0057】

レジスタについては実施の形態 1 と同様である。また、メモリ 16 (Mem A) は汎用的な 2 ポートメモリであり、メモリ 17, 18 (Mem B 1, B 2) はシングルポートメモリである。これら汎用的なメモリを用いるため、素子の IP コア化の障害が生じず、また、メモリ数の増加をシングルポートメモリを採用することによる素子面積の低減で補うことができる。

【0058】

図 6 は、図 5 の回路において擬似コード 2. x の処理を実行した時の処理のタイミングを示したタイミングチャートである。図 5 では、 $m=4$ の場合の処理の最終ループ近辺を示している。

【0059】

まず、 i ループの最初に t の計算 (擬似コード 2. 3, 2. 4) を行う。メモリ 16 から p_0 を読出し、メモリ 17 から x_3 を読出し、メモリ 18 から y_0 を読み出す。これら読出しは 1 つの読出しステージで行われる。これら読み出した

変数を用いて行番号 2. 3 の演算を行い、演算結果の下位ビット t_{mp_L} を t レジスタに入力する。前記演算と並行してメモリ 17 から N を読出し、前記 t と N を用いて行番号 2. 4 の演算を行う。演算結果の下位ビット t_{mp_L} が t レジスタに入力され、本ループ内を通じてこの t の値が用いられる。なお、先に読み出した x_3 も本ループ内を通じて用いられる。

【 0 0 6 0 】

次に j ループに入り、 p_0 , n_0 , y_0 が各々メモリ 16, 17, 18 から p_j レジスタ、 n_j , N レジスタ、 y_j レジスタに同一サイクル内で読み込まれる。その後、 c , t , x_3 の値を用いて演算が行われ、演算結果の下位ビットは p_{j-1} としてメモリ 16 に、上位ビットは c レジスタに書込まれる（行番号 2. 9, 2. 10）。なお、この j ループの $j = 0$ における処理では c は 0 リセットされる。また、 $j = 0$ では演算結果の下位ビットは記録されない。

【 0 0 6 1 】

行番号 2. 14 の処理では、 c レジスタの値が p_m としてメモリ 16 に記録される。その後の $P = P - n$, $MemA \rightarrow MemB$ の処理は実施の形態 2 と同様である。

【 0 0 6 2 】

本実施の形態によれば、2 つの乗算器を設け、さらにメモリを 3 つ設けたので、3 変数を同時に読み出すことが可能になり、2 つの乗算器を最大限に活用するデータの読出しを行うことが可能になる。

【 0 0 6 3 】

本実施の形態で必要な処理サイクルは以下の通りである。 i ループ内の t の計算で 2 サイクル、 j ループで m サイクル、擬似コード 2. 12 の処理で 1 サイクル、合計 $m + 3$ サイクルが i ループ内の処理で必要である。 i ループの m 回を乗じて $m^2 + 3m$ となる。なお $i = 0$ のときに p_j を読み出す必要はないがそのサイクルで x_i , y_j を読み出す必要があるので m を減じることはない。行番号 2. 14 の比較動作とメモリ間転送に $2m$ サイクル、パイプラインを抜けるのに 2 サイクルを要する。結局最終的なサイクル数は

$$m^2 + 5m + 2$$

となる。従来方式はもとより、実施の形態 1, 2 と比較しても大幅にサイクル数が低減される。特に m の値が大きくなると m^2 項が効いてくるので、その効果がより顕著になる。本実施の形態を用いることにより 2 乗算器の場合のメモリアクセスのボトルネックを解消でき、サイクル数を大幅に改善することが可能になる。

【0064】

以上、本発明者によってなされた発明を発明の実施の形態に基づき具体的に説明したが、本発明は前記実施の形態に限定されるものではなく、その要旨を逸脱しない範囲で種々変更可能であることは言うまでもない。たとえば、前記実施の形態では、メモリとしてシングルポートまたは 2 ポートのメモリを例示したが、3 ポートメモリにも適用できる。

【0065】

【発明の効果】

本願で開示される発明のうち、代表的なものによって得られる効果は、以下の通りである。すなわち、モンゴメリ乗算回路におけるメモリアクセスのボトルネックを 2 ポートやシングルポートの汎用メモリを使いながら解消することができる。これにより、高いパフォーマンスと素子の小型化を実現し、暗号生成回路を低いコストで提供できる。

【図面の簡単な説明】

【図 1】

本発明の一実施の形態であるモンゴメリ乗算回路の一例をそのデータパス部について示したブロック図である。

【図 2】

図 1 の回路において擬似コード 3. x の処理を実行した時の処理のタイミングを示したタイミングチャートである。

【図 3】

本発明の他の実施の形態であるモンゴメリ乗算回路の一例をそのデータパス部について示したブロック図である。

【図 4】

図 3 の回路において擬似コード 3. x の処理を実行した時の処理のタイミングを示したタイミングチャートである。

【図 5】

本発明のさらに他の実施の形態であるモンゴメリ乗算回路の一例をそのデータパス部について示したブロック図である。

【図 6】

図 5 の回路において擬似コード 2. x の処理を実行した時の処理のタイミングを示したタイミングチャートである。

【図 7】

従来方式において擬似コード 3. x の場合のモンゴメリ乗算回路データパス部の概略を示すブロック図である。

【図 8】

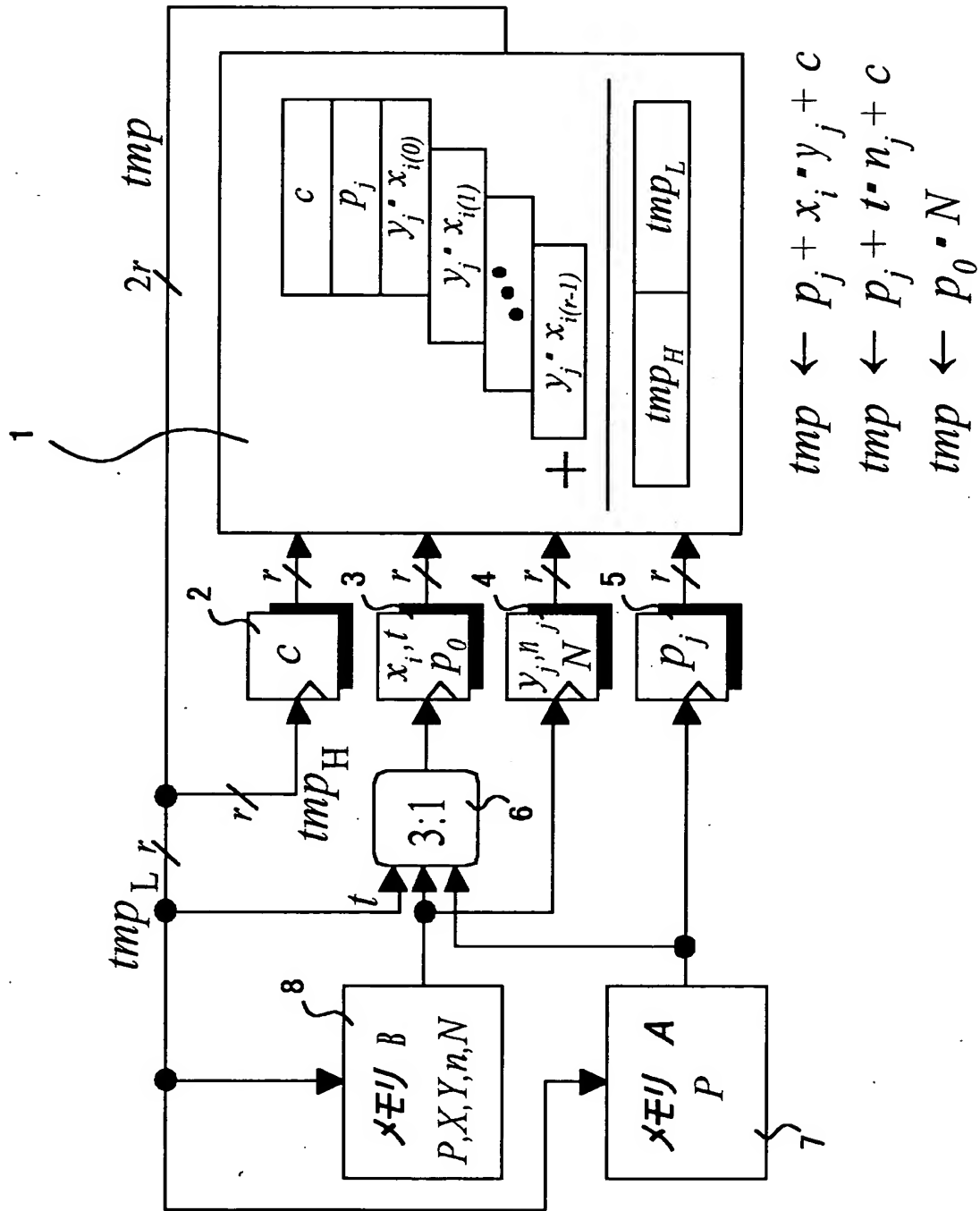
図 7 におけるブロック数 $m = 4$ の最終ループ処理近辺のタイミングチャートである。

【符号の説明】

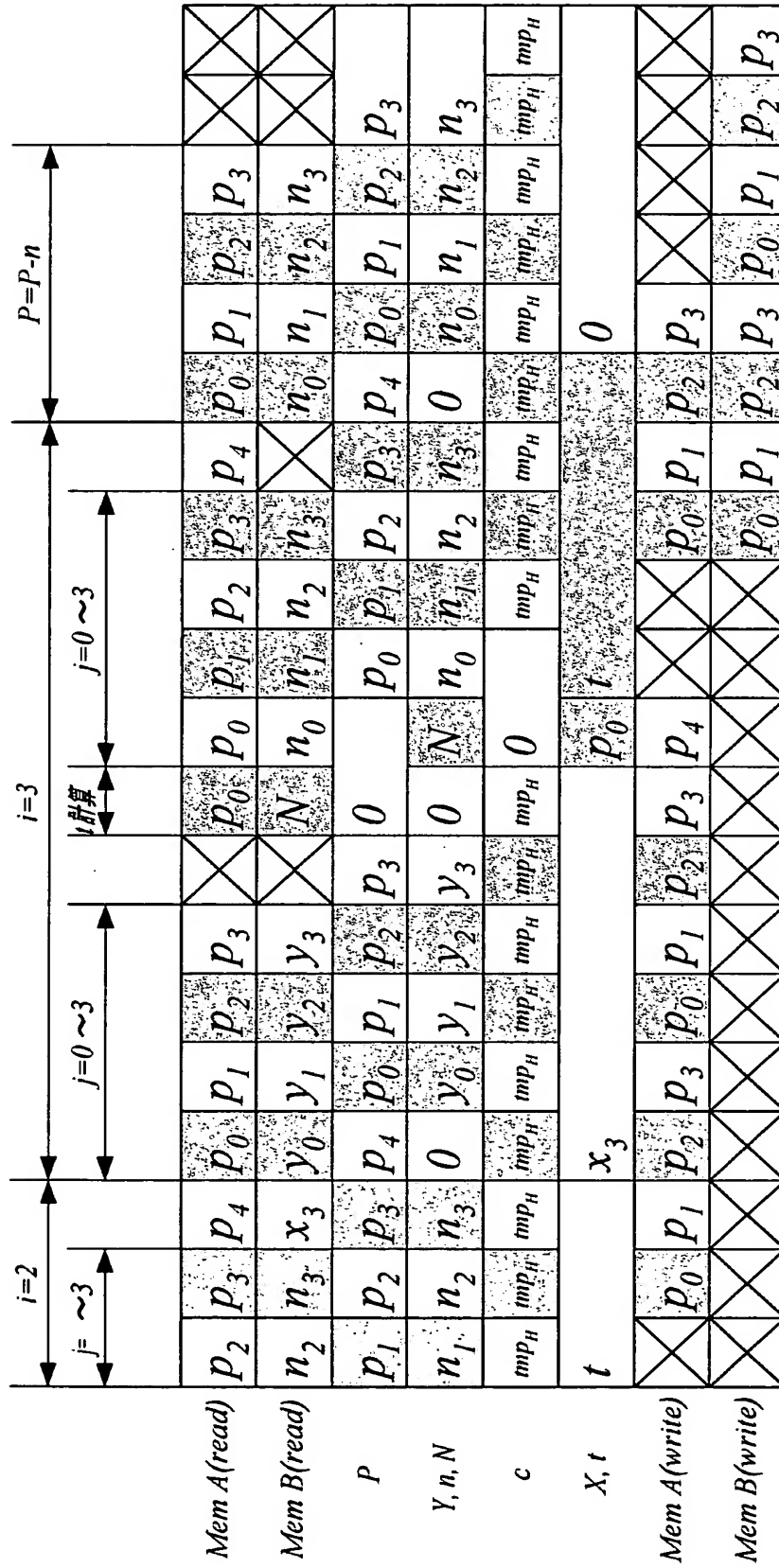
1, 9 … 積和演算回路、 2 ~ 5, 10 ~ 15 … 入力レジスタ、 6 … マルチプレクサ、 7, 8, 16 ~ 18 … メモリ。

【書類名】 図面

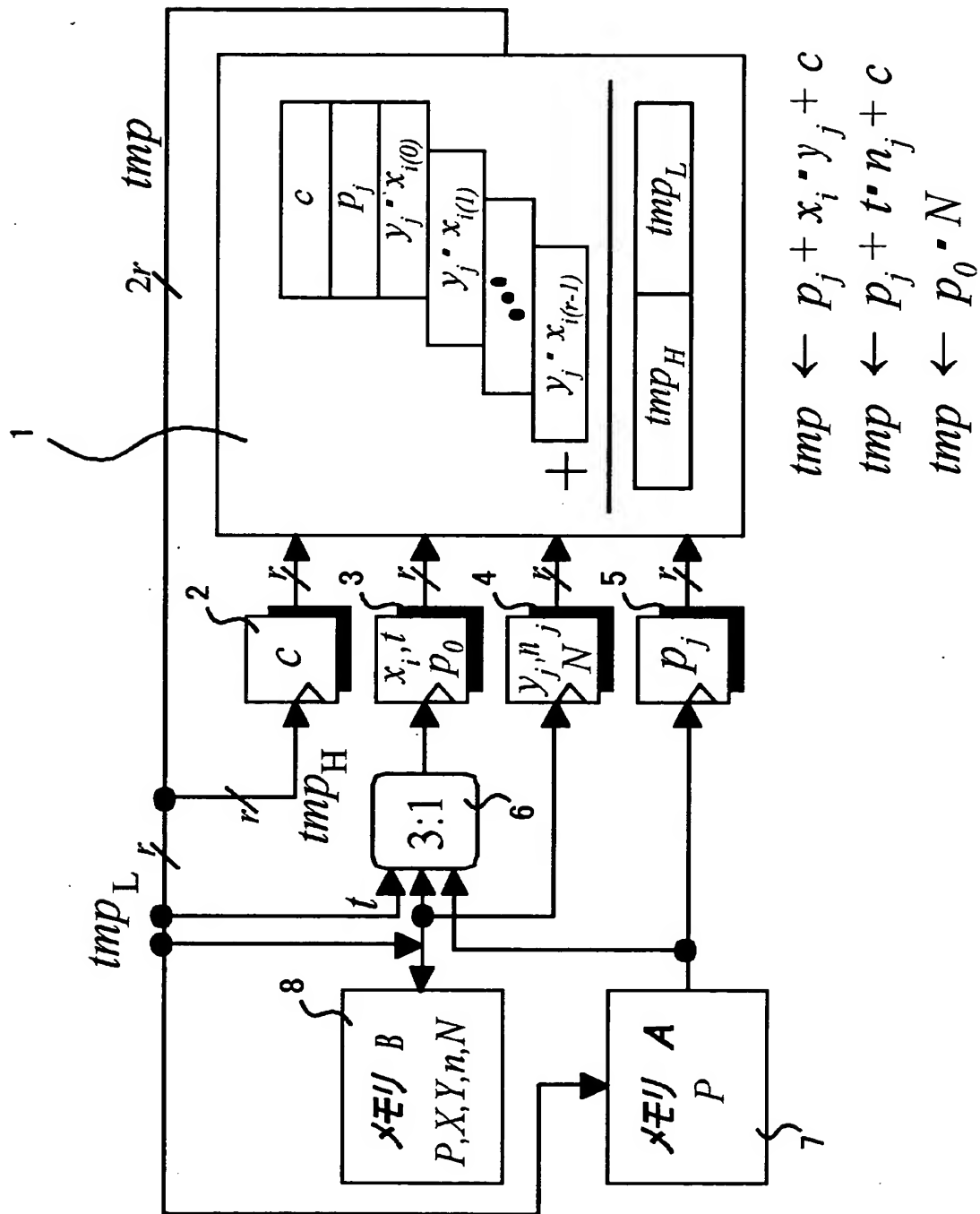
【図 1】



【図 2】



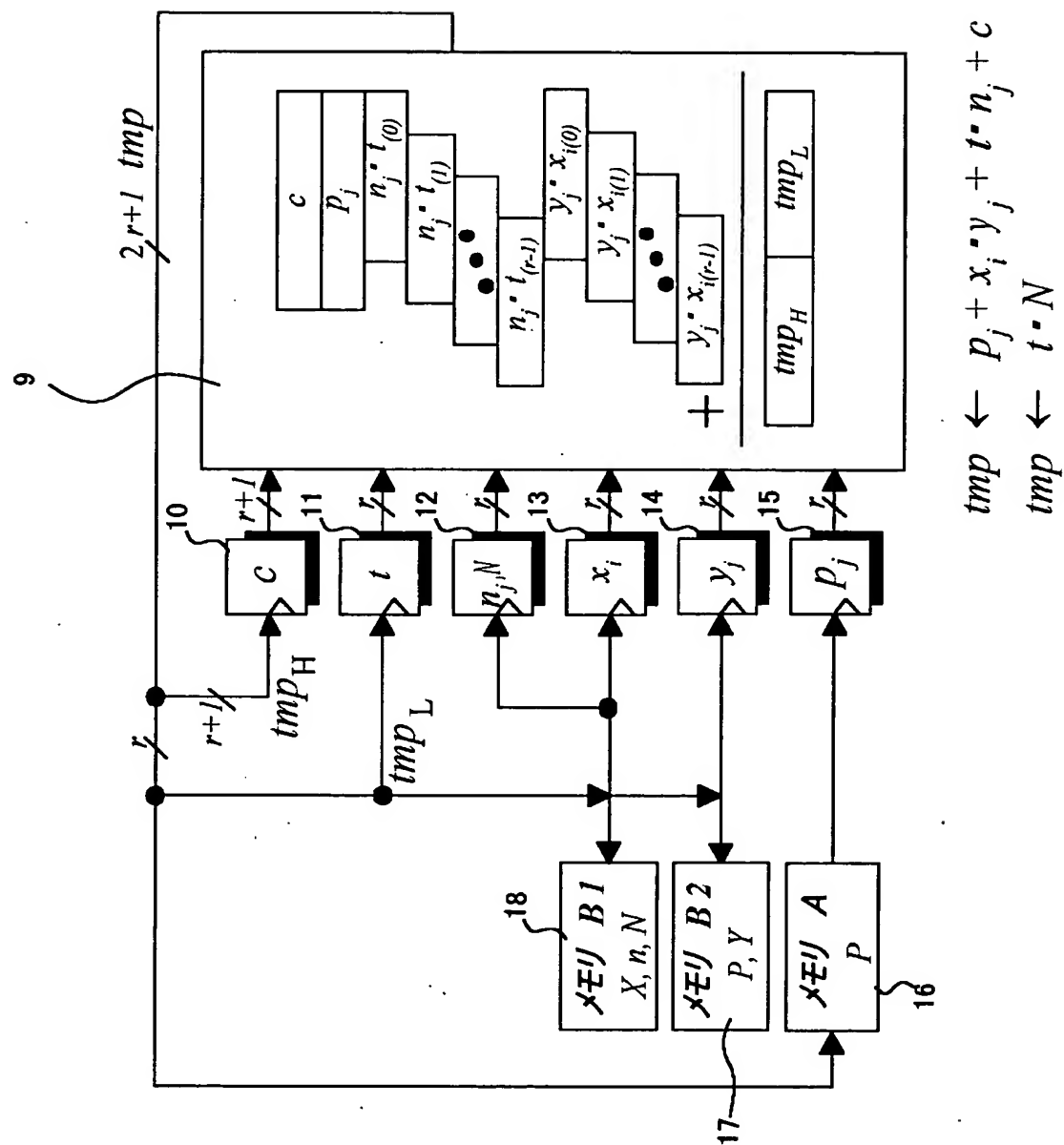
【図 3】



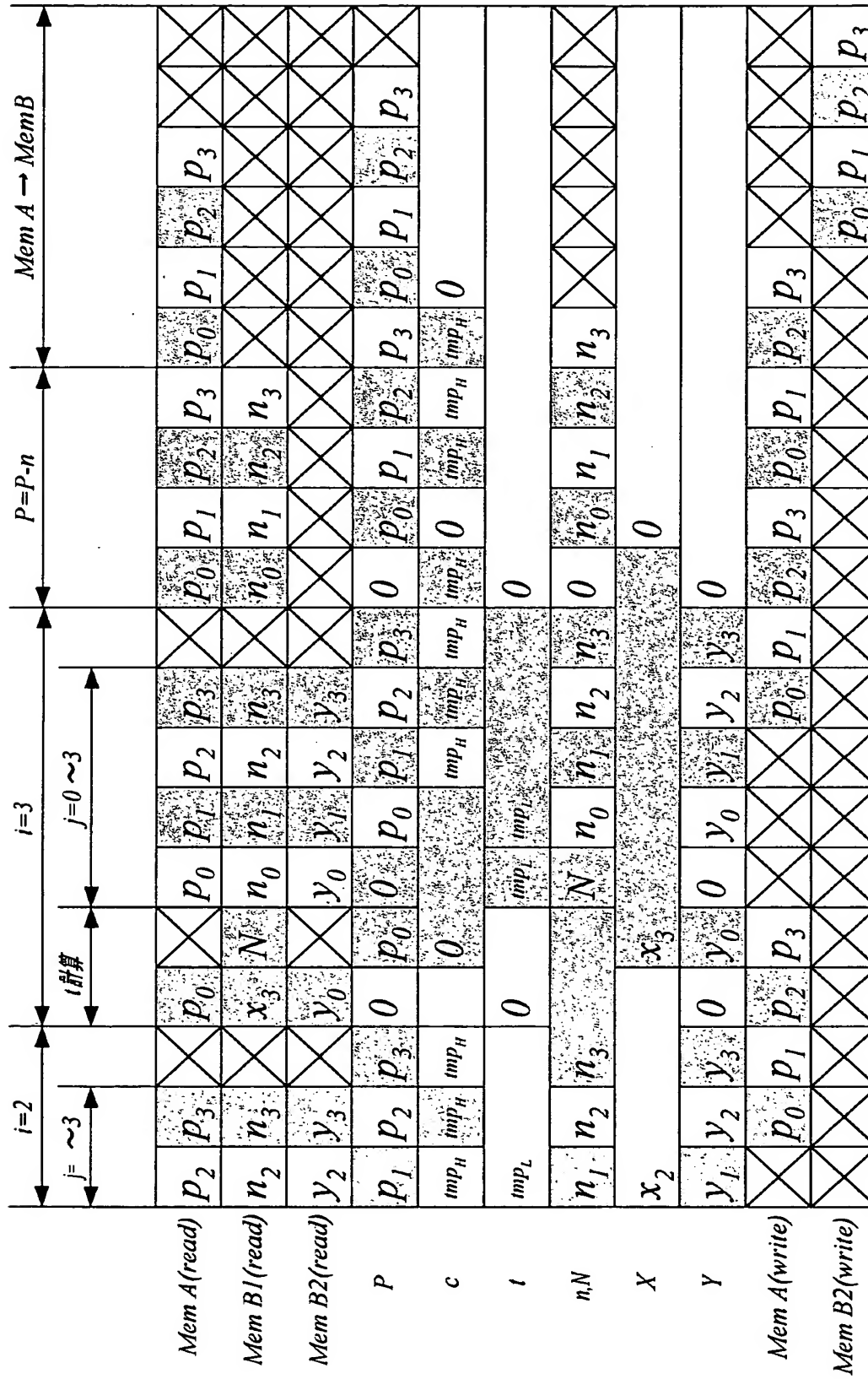
【図 4】

	$i=2$			$i=3$			$P=P-n$			$Mem A \rightarrow Mem B$		
	$j=0 \sim 3$			$j=0 \sim 3$			$j=0 \sim 3$					
Mem A(read)	p_2	p_3	p_4	p_0	p_1	p_2	p_3	p_4	p_0	p_1	p_2	p_3
Mem B(read)	n_2	n_3	x_3	y_0	y_1	y_2	y_3	p_3	n_0	n_1	n_2	n_3
P	p_1	p_2	p_3	p_4	p_0	p_1	p_2	p_3	p_4	p_0	p_1	p_2
Y, n, N	n_1	n_2	n_3	0	y_0	y_1	y_2	y_3	0	n_0	n_1	n_2
c	imp_H	imp_H	imp_H	imp_H	imp_H	imp_H	imp_H	imp_H	imp_H	imp_H	imp_H	imp_H
X, t	x_3			p_0			0			0		
Mem A(write)	p_0	p_1	p_2	p_3	p_4	p_0	p_1	p_2	p_3	p_4	p_0	p_1
Mem B(write)												

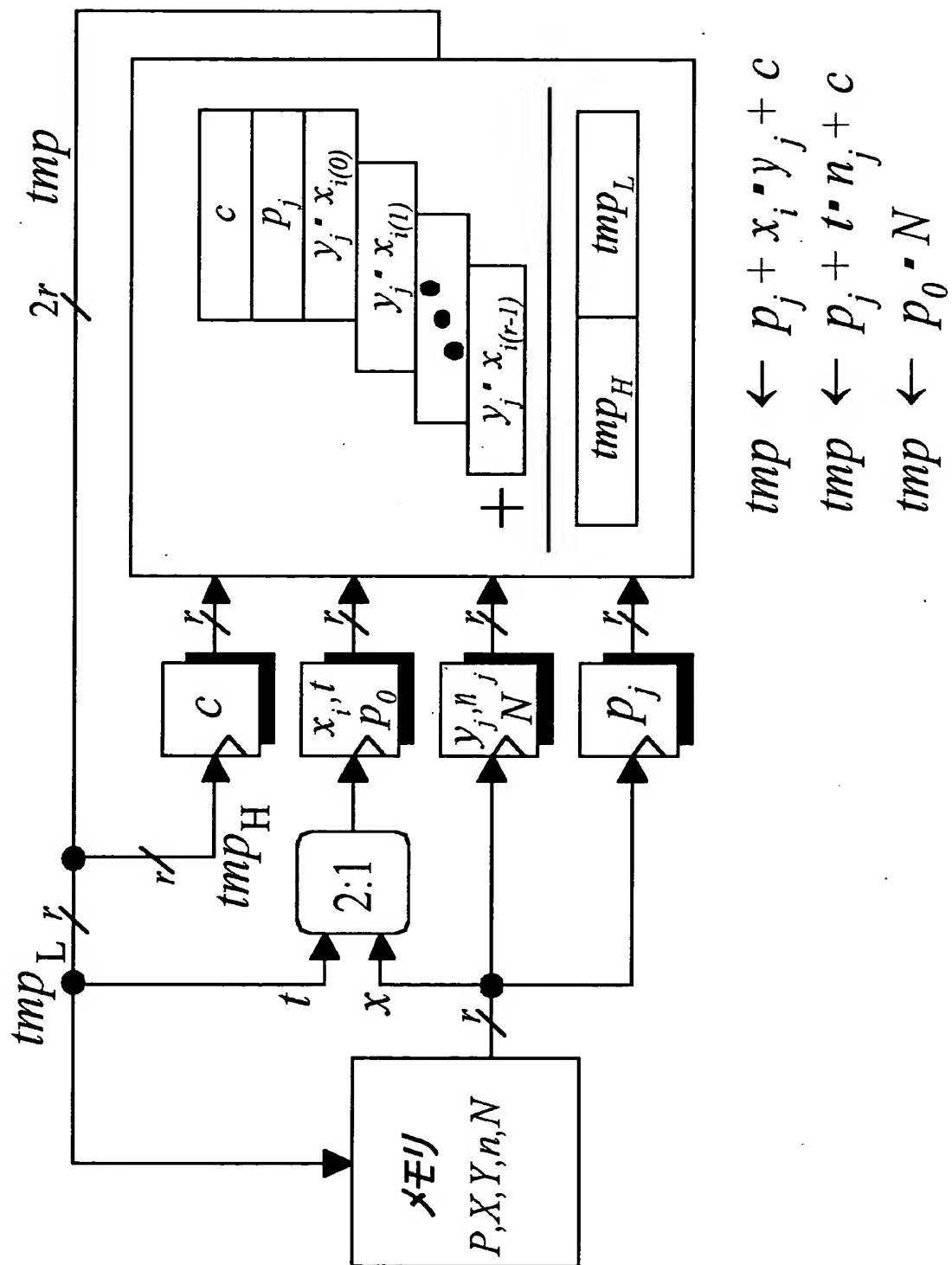
【図 5】



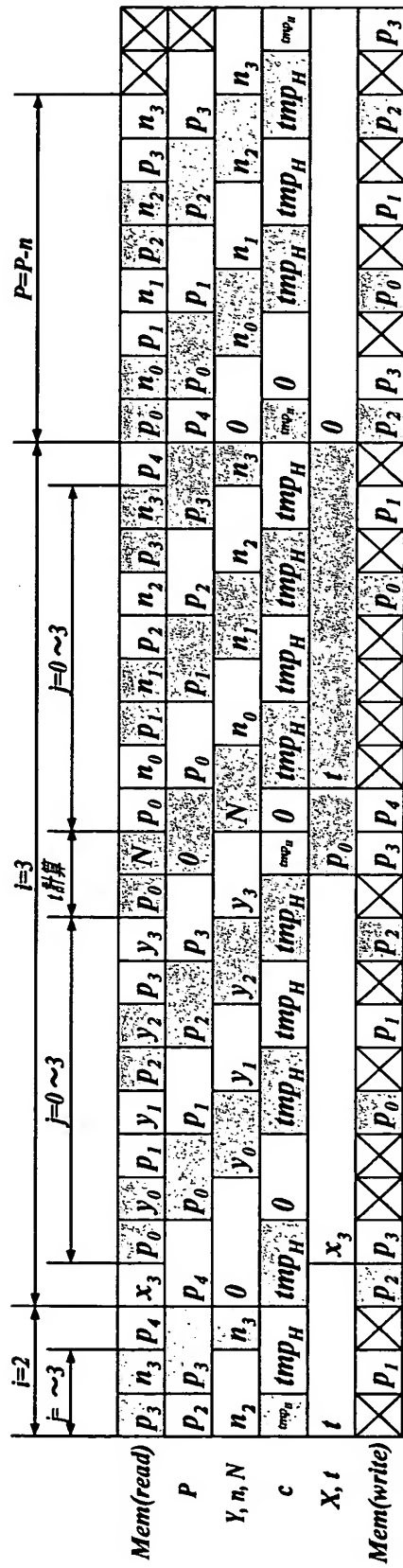
【図 6】



【図 7】



【図 8】



【書類名】 要約書

【要約】

【課題】 モンゴメリ乗算回路におけるメモリアクセスのボトルネックを2ポートやシングルポートの汎用メモリを使いながら解消する。

【解決手段】 2つのメモリ7、8を設け、演算に必要な変数のうちメモリから読み出す必要のある変数については異なるメモリに記録する。そしてパイプライン処理の同一の読出しステージでメモリ7からレジスタ5に変数を読出し、メモリ8からその他のレジスタに他の変数を読み出す。

【選択図】 図1

認定・付加情報

特許出願の番号	特願 2 0 0 0 - 3 8 6 0 6 9
受付番号	5 0 0 0 1 6 3 9 4 0 4
書類名	特許願
担当官	濱谷 よし子 1 6 1 4
作成日	平成 1 3 年 2 月 8 日

<認定情報・付加情報>

【特許出願人】

【識別番号】	390009531
【住所又は居所】	アメリカ合衆国 1 0 5 0 4、ニューヨーク州 アーモンク (番地なし)
【氏名又は名称】	インターナショナル・ビジネス・マシーンズ・コーポレーション

【代理人】

【識別番号】	100086243
【住所又は居所】	神奈川県大和市下鶴間 1 6 2 3 番地 1 4 日本アイ・ビー・エム株式会社 大和事業所内
【氏名又は名称】	坂口 博

【代理人】

【識別番号】	100091568
【住所又は居所】	神奈川県大和市下鶴間 1 6 2 3 番地 1 4 日本アイ・ビー・エム株式会社 大和事業所内
【氏名又は名称】	市位 嘉宏

【代理人】

【識別番号】	100106699
【住所又は居所】	神奈川県大和市下鶴間 1 6 2 3 番 1 4 日本アイ・ビー・エム株式会社大和事業所内
【氏名又は名称】	渡部 弘道

【復代理人】

申請人	
【識別番号】	100112520
【住所又は居所】	神奈川県大和市中心林間 3 丁目 4 番 4 号 サクライビル 4 階 間山・林合同技術特許事務所
【氏名又は名称】	林 茂則

【選任した復代理人】

【識別番号】	100110607
--------	-----------

次頁有

認定・付加情報（続き）

【住所又は居所】	神奈川県大和市中央林間3丁目4番4号 サクラ イビル4階 間山・林合同技術特許事務所
【氏名又は名称】	間山 進也

出 願 人 履 歴 情 報

識別番号 [390009531]

1. 変更年月日 2000年 5月16日

[変更理由] 名称変更

住 所 アメリカ合衆国10504、ニューヨーク州 アーモンク (番地なし)

氏 名 インターナショナル・ビジネス・マシーンズ・コーポレーション